Topic Modelling

Enhance Expert Search

- Problem Statement:
  - We wanted to enhance the expert search system with the topic modelling of the BIOS . So that we could tag search results with top relevant skills of each BIOS.
- Steps Taken:
  - We decided to use spaCy library to do topic modeling on bio pages.
  - We ran the topic modeler on a small selection of bio pages prepared from the facility page classifier and we were able to verify manually verify whether the topic modeler works correctly.
  - We stored the data coming out of the spaCy code into TinyDB. So, we could query the results of top relevant skills based on file name whenever needed.

Technology Used:

- Spacy is written in cython language, (C extension of Python designed to give C like performance to the python program). Hence is a quite fast library. spaCy provides a concise API to access its methods and properties.
- We used Part-of-Speech (POS) Tagging, Named Entity Recognition and tokenization features of spacy
- Tokenisation is a foundational step in spacy library. Tokenising text helped us splitting a piece of text into words, symbols, punctuation, spaces and other elements, thereby creating “tokens” which could be used in POS tagging and Named Entity Recoginations.
- POS tagging is the task of automatically assigning Part of speech tags to all the words of a sentence. It is helpful in various downstream tasks in Natural language processing for removing the unwanted and noisy words.
- Named Entity Recognition feature in the library has helped us find the common things such as persons, locations, organizations, etc. and segregate along with the noise words in BIOS.
- TinyDB is a lightweight document oriented database and we have used it in saving our final results of Topic Modelling in Json format in TinyDB. So we could query the table whenever needed with the file name and find the keywords specific to the file.

Code Snippet:

Some Sample Code Snippet is show below for reference:

Code for removing noise word before finding the keywords :

```
def isNoise(token):
    is_noise = False
    if token.pos_ in noisy_pos_tags:
        is_noise = True
    elif token.is_stop == True:
        is_noise = True
    elif len(token.string) <= min_token_length:
        is_noise = True
    elif token.string.lower().strip() in stop_words:
        is_noise = True
    elif token.string.strip() in ents:
        is_noise = True
    return is_noise
```

Code for finding named entity recognization

```
ents = [e.text for e in document.ents]
```

Code for finding top key words :

```
from collections import Counter
cleaned_list = [cleanup(word.string) for word in document if not isNoise(word)]
counts=Counter(cleaned_list).most_common(5)
```

Code for Saving data into TinyDB :

```
for key in counts:
    a.append(key[0])

db = TinyDB('Topic_Model_Results.json')
db.insert({'File_Name': i, 'Topic_Modelled_word': a})
```

Sample Output:

- Below is sample file which is topic modelled.

Home Bio Research Publications Research Group Contact Info Engineering at Illinois Vikram S. Adve Interim Head and Donald B. Gillies Professor|Computer Science Department|University of Illinois at Urbana-Champaign Research Projects Prof. Vikram Adve F ollow the individual project links for more details about each project and relevant publications: ALLVM: Exploring the benefits for software performance, security and reliability if all software on a system (either all userspace software or or userspace+OS software) is available in a rich virtual instruction set that can be analyzed and transformed by sophisticated compiler techniques (think Java bytecode, but for all software). Heterogeneous Parallel Virtual Machine : A compiler infrastructure and parallel program representationfor heterogeneous parallel systems, with the goal of making it much easier to write performance-portable parallel programs. A single program in the HPVM representation can be compiled to GPUsand tomulticore CPUs (with and without) vector extensions, while achieving performance close to separately hand-tuned code for each of those systems. The project is alsoexploring code generation for FPGAs, for specializeddeep-in-memory compute hardware, and developing optimizingcompilersforparallel languages like OpenMP and Domain Specific Languages. Automated Debugging for Software Failures : We are developing automated static and dynamic analysis techniques tounderstand the causes of failures in software systems, in order tohelp programmers diagnose and fix software bugs with as little effort as possible. The project is investigating automated fault localization and diagnosis techniques for both standalone and distributed

You can see he is part of research team and his interest is on compilers.

Output from the model:

```
from tinydb import TinyDB, Query
db = TinyDB('Topic_Modelling.json')
result = db.get(Query()['File_Name'] == '2.txt')
print(result)
```

```
{'File_Name': '2.txt', 'Topic_Modelled_word': ['research', 'compiler', 'software', 'students', 'projects']}
```

Challenges and Links

Challenges and Resolution :

One of the challenges which we faced was to setup the expert search system working in local. Our initial goal was to have the topic modelled words display on the webpage. We tried different options and as we could not do that, we transformed our code to save the results in TinyDB and retrieve data whenever needed. Setup instructions of the DB has been added in the GIT read me page.

Links:

Read me Page and set up instruction : https://github.com/karthikrajagopal87/Topic-Modelling-Spacy
Presentation Link: https://github.com/karthikrajagopal87/Topic-Modelling-Spacy/blob/main/Topic_Modelling_Presentation.pdf
Tiny DB output : https://github.com/karthikrajagopal87/Topic-Modelling-Spacy/blob/main/Topic_Modelling.json
Spacy Code : https://github.com/karthikrajagopal87/Topic-Modelling-Spacy/blob/main/spacy_code.py
Sample BIO Pages tested : https://github.com/karthikrajagopal87/Topic-Modelling-Spacy/tree/main/bios